UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO:

VISION-BASED AUTONOMOUS LANDING SYSTEM FOR AERIAL VEHICLES

Autor:

Javier Ortega Bueno

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Tutor: Abdulla Hussein Abdulrahman Al-Kaff

Director: José María Armingol Moreno

Departamento de Ingeniería de Sistemas y Automática

Junio 2014

AGRADECIMIENTOS

A mis padres,

porque sin ellos recorrer todo este camino hubiera sido imposible.

RESUMEN

En este trabajo de fin de grado se tiene como fin el diseño de un sistema de aterrizaje autónomo, basando en odometría visual, para el vehículo aéreo no tripulado (UAV) Ardrone 2.0 de Parrot. Esta idea surge a partir de un proyecto del Departamento de Sistemas Inteligentes de la Universidad Carlos III de Madrid que está creando una aplicación que permita convertir este aparato en un sistema aéreo no tripulado (UAS), es decir, un drone con capacidad de realizar maniobras autónomas.

Este proyecto se basa en crear un modulo de ampliación para esta aplicación, que permita al Ardrone aproximarse y aterrizar sobre una zona delimitada para tal efecto, basándose solo en la información obtenida a través de las cámaras integradas.

Se utilizaran técnicas de visión por computador apoyadas en la plataforma Emgu CV, que a su vez está basada en las conocidas librerías de OpenCV. La navegación y aterrizaje se realizará con el envió de los comandos de navegación en tiempo real basados en la información obtenida al realizar el análisis de la imagen y la búsqueda del patrón predefinido.

ABSTRACT

In this final degree project the objective is to design an autonomous landing system, basing on visual odometry, for an unmanned aerial vehicle (UAV) Parrot Ardrone 2.0. This idea comes from a project of the Department of Intelligent Systems in the University Carlos III de Madrid who are creating an application that allows converting this aircraft in an unmanned aerial system (UAS), in essence, a drone with the capacity to perform autonomous manoeuvres.

This project builds on creating an expansion module for this application, allowing the Ardrone approaching and landing on a limited area for this purpose, based only on the information obtained from the integrated cameras.

Vision by computer techniques are used supported in CV Emgu platform, which is based on the famous OpenCV library. Navigation and landing is performed with commands sent in real time navigation based on the information obtained from the image analysis and the search for predefined pattern.

INDICE GENERAL

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

ABREVIATURAS

UAV: Unmanned Aerial Vehicle, vehículo aéreo no tripulado.

UAS: Unmanned Aircraft System, vehículo aéreo autónomo.

HALE: High Altitude Long Endurance, largo rango y gran duracion.

MALE: Medium Altitude Long Endurance, medio rango y gran duracion.

UCAV: Unmmaned Combat Air Vehicle, vehículo aéreo no tripulado de combate.

DARPA: Defense Advanced Research Projects Agency.

SDK: Kit de Desarrollo de Software.

AUVSI: Asociación Internacional de Vehículos No Tripulados.

API: Interfaz de programación de aplicaciones, del inglés Application Programming Interface.

SURF: Speeded Up Robust Features.

SIFT: Scale Invariant Feature Transform.

DoG: Difference of Gaussian, diferencia de gaussianas.

ROI: Region of interest.

Capítulo 1: INTRODUCCION

1.1 Motivación

Hoy en día el uso de vehículos aéreos no tripulados (en inglés responde a las siglas UAV, Unmanned Aerial Vehicle) en configuración de quadrotor se ha hecho muy popular en el campo de la investigación y desarrollo de nuevos productos.

Los principales motivos de esta creciente popularidad son la gran estabilidad y versatilidad que aporta el disponer de cuatro rotores además del bajo coste de los aparatos, como el que utilizaremos para este trabajo.

El Laboratorio de Sistemas Inteligentes de la universidad Carlos III de Madrid está desarrollando una aplicación para el Ardrone que incluye características de navegación autónoma mediante odometría visual, de ahí nace la necesidad de disponer de un sistema que permita realizar un aterrizaje autónomo en la zona indicada para tal efecto.

La motivación de realizar este trabajo se basa en un interés tanto personal como profesional de aprender en cuanto a UAV y sistemas de visión por computador.

1.2 Objetivos y enfoque

El objetivo del proyecto es desarrollar un sistema de aterrizaje autónomo para un UAV mediante la búsqueda de un patrón a través de las imágenes obtenidas por la cámara integrada en el quadrotor.

Se utilizaran técnicas de visión por computador vistas en [1] y en la asignatura de Sistemas de Percepción. La navegación y aterrizaje se realizará con el envió de los comandos de navegación en tiempo real basados en la información obtenida al realizar el análisis de la imagen. De esta manera se conseguirá un aterrizaje totalmente autónomo en una zona dispuesta especialmente para ello.

1.3 Elección y descripción del hardware

El software elegido para la realización de este proyecto es el Parrot Ardrone 2.0, ya que, por su bajo coste y su buena accesibilidad (se distribuye en cualquier comercio especializado o gran superficie, además de la gran disponibilidad de repuestos) se ajusta perfectamente a los requerimientos de este trabajo.

Existe una primera versión, el Ardrone 1.0, con la que existen ligeras diferencias. La versión 2.0 contiene mejoras en cuanto a la cámara frontal, pasa de VGA (640 x 480) a resolución HD (1280 x 720), y en cuanto al número de sensores y la calidad de los mismos.

Ilustración 1: Software elegido Ardrone 2.0

Capítulo 2: ESTADO DEL ARTE

2.1 Tecnología UAV

Hasta pocos años desarrollar tecnologías relacionadas con UAV era muy raro, debido al alto coste que suponía el construir este tipo de aparatos. El origen de este tipo de vehículos tuvo fines militares, con el objetivo tanto de vigilar una zona, como de realizar misiones de ataque sin poner en peligro vidas humanas.

En la actualidad se ha reducido mucho el coste de los componentes necesarios tales como sensores, motores y cámaras, por lo que en estos momentos el desarrollo de tecnologías relacionadas con UAV está en pleno auge.

En los últimos años, los quadrotors se han popularizado y se han convertido en una línea de investigación a seguir a causa de las numerosas ventajas que presentan.

En función del objetivo con el que se diseñe, el UAV tendrá una tecnología y unas dimensiones adecuadas para cumplir con eficiencia la tarea para la que ha sido diseñado. El hablar de que el vehículo es no tripulado puede dar lugar a confusiones, es cierto que en el interior del vehículo no se encuentra ningún operador, pero existe un contacto entre el UAV y la estación de tierra, donde se monitoriza o se pilota la aeronave.

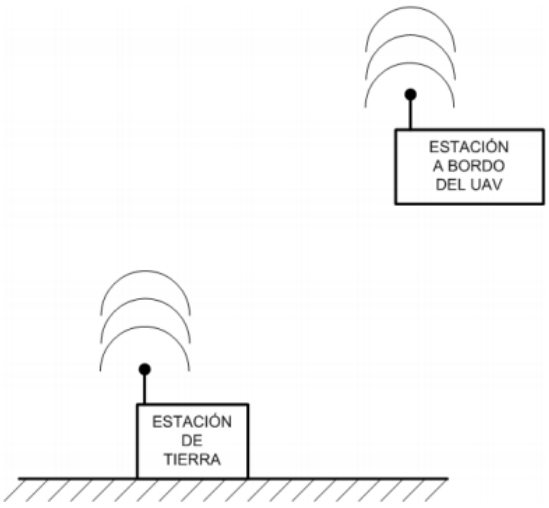

Ilustración 2: Estaciones de un UAV

Una vez que se superó el reto de poder manejar un vehículo no tripulado, la investigación continúo al siguiente nivel, lo que se conoce como UAS (*Unmanned Aerial System* en inglés). La principal diferencia reside en que un UAS es un vehículo que cuenta con un sistema autónomo para seguir una trayectoria ya predefinida y que puede estar preparado para tomar decisiones autónomas en base a la información obtenida por sus propios sensores.

También cabe resaltar que no todo vehículo no tripulado que pueda volar se puede considerar UAV o UAS. Para ser considerado como tal es necesario que estas aeronaves no sean desechables y puedan realizar un vuelo de retorno. Por ejemplo, un misil no puede ser considerado como un UAS, aunque posea tecnología propia de estos.

2.1.1 Clasificación de los UAV

Existe una gran variedad de modelos de UAV o UAS, fabricados tanto con fines civiles, recreativos o militares, siendo estos últimos los más numerosos. En [2] podemos apreciar la infinidad de modelos militares existentes. Los tipos de aeronaves son aviones, dirigibles y una gran variedad de configuraciones diferentes de helicópteros. Se podrían hacer muchas clasificaciones, pero las más importantes están basadas en el tamaño que va estrechamente relacionado con la finalidad para la que han sido concebidos.

Los UAV más grandes son los llamados HALE. Diseñados para poder volar por encima de los 20.000 metros, quedando por encima de las rutas comerciales, con un alcance cercano a los 10.000 kilómetros y una autonomía de 24-48 horas. Los HALE están diseñados para desempeñar misiones estratégicas de vigilancia militar. La propulsión de este tipo de UAV lo normal es que esté proporcionada por motores a reacción. El UAV más conocido de esta clase es el Northrop Grumman RQ-4 Global Hawk, propiedad de los EEUU, con sus 35,41 m de envergadura, es capaz de vigilar y proporcionar información de 103.600 km² de terreno por día, una superficie

equivalente a la de Portugal. Dispone de un radar de apertura sintética que le permite vigilar a través de zonas cubiertas de nubes o incluso tormentas de arena, además también lleva integrado multitud de equipos ópticos e infrarrojos. [3] [4]

Ilustración 3: Northrop Grumman RQ-4 Global Hawk

En la categoría de UAV de tamaño medio nos encontramos con los MALE, que han sido diseñados para tener una gran rango y larga autonomía pero volando a altitudes medias, entre los 5000 y 8000 metros. Este tipo de aeronaves pueden ir propulsadas tanto por motores a reacción como por tubo hélices o motores de pistón. Tienen una gran variedad de utilidades, tanto civiles como militares. Pueden servir tanto como aviones de vigilancia o de ataque.

Como avión de vigilancia y ataque nos encontramos con el UAV más famoso y polémico hasta momento, el General Atomics MQ-1 Predator, con 8,23 metros de envergadura y propulsado por un motor a pistón de cuatro cilindros. Actualmente operando en las fuerzas armadas de EEUU, Reino Unido e Italia. Está concebido principalmente para tareas de reconocimiento pero además tiene capacidad ofensiva, pudiendo incorporarle misiles de gran potencia aire-tierra o aire-aire. [5] [6]

Ilustración 4: General Atomics MQ-1 Predator

También dentro de la categoría MALE, podemos encontrar aviones puramente de combate (UCAV), estos todavía se encuentran en fase experimental. Un ejemplo de ello es el EADS Barracuda, desarrollado conjuntamente por España y Alemania, tiene una envergadura de 7,22 metros y esta propulsado por un motor a reacción turbofan. Este proyecto fue desarrollado como demostrador de tecnología para el desarrollo futuro de sistemas de misiones no tripuladas de combate. [7] [8]

Ilustración 5: EADS Barracuda

Dentro de la categoría MALE nos podemos encontrar otro tipo de configuraciones diferentes de aeronave, como es el Northrop Grumman MQ-8 Fire Scout, un helicóptero militar usado para misiones de reconocimiento y ataques de alta

precisión. Tiene una autonomía de más de 5 horas y un rango de combate de 200 kilómetros. [9] [10]

Ilustración 6: Northrop Grumman MQ-8 Fire Scout

Los pequeños UAV o SUAV, son aeronaves suficientemente pequeñas para poder ser trasportadas por un humano. Estos están diseñados para tareas de reconocimiento, vigilancia y para usos recreativos.

Para tareas de vigilancia y reconocimiento, el más novedoso es el Honeywell RQ-16A T-Hawk, desarrollado por Honeywell. Este es un pequeño UAV con un solo ventilador situado en su centro y propulsado por un pequeño motor de gasolina. Su peso ronda los 8 kilogramos, lo cual le proporciona una autonomía de 40 minutos y una velocidad máxima de 130 km/h.

Su creación surge a través de la agencia DARPA, que es una agencia del Departamento de Defensa de Estados Unidos responsable del desarrollo de nuevas tecnologías para uso militar y actualmente es usado por las fuerzas armadas de varios países en territorios en guerra. [11]

Ilustración 7: Honeywell RQ-16A T-Hawk

Este gran desarrollo de pequeños vehículos no tripulados con fin militar, como el nombrado RQ-16 T-Hawk, trae la aparición de multitud de modelos con un fin recreativo con una gran variedad de configuraciones aerodinámicas y de propulsión.

La aparición del Ardrone 1.0 de Parrot, que es la empresa francesa que se encarga de su fabricación y su distribución, supone una gran revolución en esta industria de UAV recreativos.

Comparte una configuración en cuadricóptero con motores eléctricos con otros modelos comerciales, pero su diferenciación y donde reside su gran éxito es en la incorporación de un microprocesador de altas prestaciones capaz de procesar una gran cantidad de señales que le llegan a través de una serie de sensores, junto con una revolución en su estación de control, ya que, permite que el Ardrone pueda ser manejado desde un Smartphone convencional.

Ilustración 8: Ardrone 1.0

Este modelo incorpora dos cámaras que le permiten capturar el entorno y una conexión Wi-Fi que es la que permite la conexión con la estación de control.

Poco después Parrot lanzó la versión 2.0 de su Ardrone. Esta incorpora mejoras en casi todas sus características, pero mantiene la esencia de su conexión Wi-Fi y su control a través de un Smartphone.

Ilustración 9: Ardrone 2.0

La llegada de este nuevo modelo, su precio asequible y las facilidades que ofrece Parrot con la distribución gratuita de su SDK, ha hecho que Ardrone se popularice en los laboratorios de tratamiento de imágenes de todas las universidades.

2.2 Visión por computador

En la actualidad, las cámaras no solo se usan para tomar imágenes o videos, ya que, el campo de la visión por computador ha avanzado mucho en estos últimos años, lo cual, nos permite usar las cámaras como si un sensor mas se tratara a la hora de extraer información del entorno.

La elección de la colocación de las cámaras siempre supone un quebradero de cabeza para los investigadores, puesto que, esta elección limitará el área de visión de la cámara. Otro punto importante a tener en cuenta es la selección del tipo que cámara a utilizar. Estas pueden ser a color, monocromáticas, infrarrojas, térmica, de visión estéreo, etc.

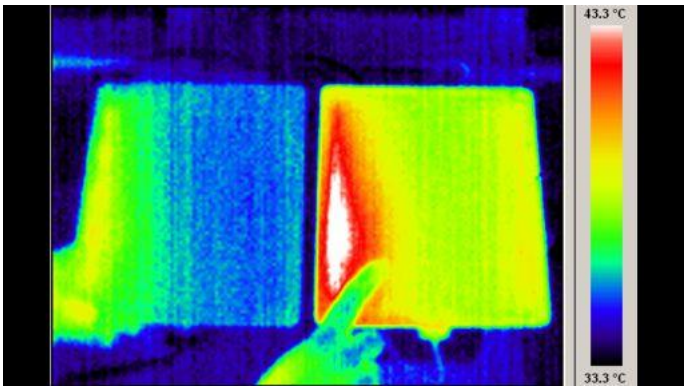

Ilustración 10: Imagen de cámara térmica

Las cámaras pueden ir integradas en el propio vehículo (*camera on board*) o pueden estar distribuidas y fijas por la zona de operación. La opción de cámaras fijas en el área de operación solo será factible para entornos de laboratorios de pruebas o académicos donde se vayan a realizar ensayos.

El Institute for Dynamic Systems and Control de Zurich, Suiza, ha desarrollado un gran trabajo a través de una sala con una instalación de cámaras fijas que permiten obtener una gran precisión del vuelo de unos cuadricóptero. De esta manera pueden crear algoritmos para que vehículo realice maniobras complejas a gran velocidad como realizar acrobacias manteniendo en equilibrio una barra de gran longitud.

Ilustración 11: Acrobacias con barra

También existen multitud de trabajos del Institute for Dynamic Systems and Control [12] para la navegación en formación mediante algoritmos de control visual, por ejemplo han desarrollado un algoritmo para manejo conjunto por tres drones de una red para lanzar y recoger una pelota.

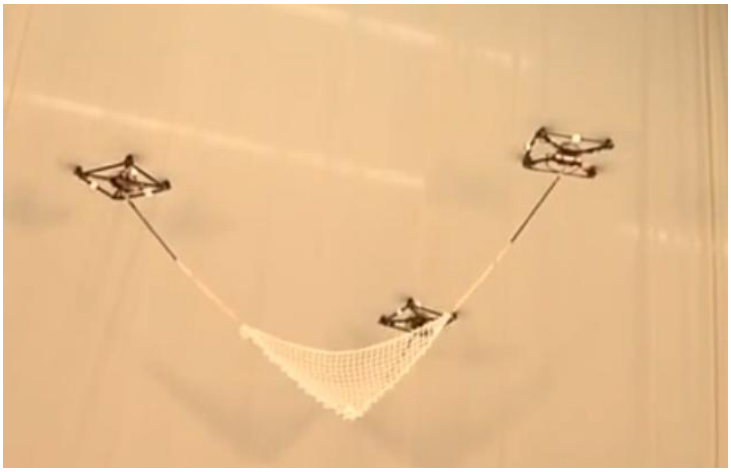

Ilustración 12: Realización de tareas conjuntas

En muchas áreas de investigación también se utiliza el sistema de Microsoft Kinect, este dispositivo cuenta con una cámara RGB, un sensor de profundidad que nos permiten obtener mucha información pudiendo desarrollar sistemas de control gestual muy intuitivos.

Ilustración 13: Microsoft Kinect

Por último, cabe destacar las librerías de programación visual OpenCV, las cuales integran una gran variedad de algoritmos de alto nivel para el tratamiento digital de la imagen así como para la visión por computador. [13] [14]

2.3 Normativa UAV

Actualmente se está trabajando para decretar una normativa oficial sobre los vehículos aéreos no tripulados. Actualmente en España, cualquier operación con UAV requiere autorización específica por parte de las autoridades aeronáuticas competentes.

Actualmente la Unión Europea está trabajando para que los UAS de gran tamaño puedan utilizar el espacio aéreo civil. Esto supondría abrir una puerta a que los vuelos comerciales puedan ser no tripulados.

La Asociación Internacional de Vehículos No Tripulados [15], AUVSI, dicta unas normas de conducta con el objetivo de concebir seguridad y acelerar la confianza pública en estos sistemas.

Las normas de la AUVSI son:

Seguridad

• No operaremos UAS de forma que represente riesgo para las personas o propiedades en la superficie o en el aire.

• Aseguraremos que los UAS serán pilotados por personas que están debidamente entrenadas y tienen competencias para operar vehículos o sus sistemas.

• Aseguraremos que los vuelos de los UAS serán realizados solo después de una rigurosa valoración de los riesgos asociados con la actividad. Esta valoración de riesgos, incluirá, pero no limitará.

• Condiciones climáticas en relación con la capacidad de los sistemas.

Identificación de normalidad de modos de fallo anticipado (perdida de enlace, fallos de potencia, pérdida de control, etc.) y las consecuencias de los fallos.

• Condiciones físicas de la tripulación para operar el vuelo.

• Cumplimiento de las regulaciones de aviación y de la apropiada operación en el espacio aéreo y procedimientos no nominales.

• Comunicación, comando, control y requisitos de espectro de frecuencia del palead (carga útil).

• Fiabilidad, rendimiento y aeronavegabilidad mediante estándares establecidos.

Profesionalidad

• Cumpliremos con las leyes nacionales, estatales y locales, ordenanzas, acuerdos y restricciones relativos a las operaciones de UAS.

• Operaremos nuestros sistemas como miembros responsables de la comunidad aeronáutica.

• Seremos sensibles a las necesidades de las personas.

• Cooperaremos totalmente con las autoridades nacionales, regionales y locales en el despliegue en respuesta a emergencias, investigación de accidentes y relaciones con los medios.

• Estableceremos planes de contingencia para todos los eventos anticipados y los compartiremos abiertamente con todas las autoridades pertinentes.

Respeto

• Respetaremos los derechos de otros usuarios en el espacio aéreo.

• Respetaremos la privacidad de las personas.

• Respetaremos los asuntos del público así como los relativos a las operaciones de los UAS.

• Apoyaremos la mejora del conocimiento y educación pública sobre las operaciones de los UAS.

Capítulo 3: HARDWARE Y SOFTWARE DEL SISTEMA

3.1 Hardware

El dispositivo elegido para llevar a cabo este proyecto es el Ardrone 2.0. Se trata de un vehículo aéreo no tripulado radiocontrolado diseñado para un uso recreativo civil diseñado y distribuido por la empresa francesa Parrot. Esta propulsado por cuatro motores eléctricos en configuración de cuadricóptero. La primera versión de este producto fue presentada y lanzada al mercado en Las Vegas International Consumer Electronics Show del año 2010. Tras una exhaustiva revisión y actualización del producto, en el mismo evento del año 2012, Parrot lanzo al mercado el Ardrone 2.0.

Este producto con fin recreativo es similar a otras aeronaves del mercado creadas con el mismo propósito, pero el producto de Parrot se diferencia del resto en que incorpora una cantidad elevada de sensores y un microprocesador potente, capaz de trabajar con los datos que recoge de los numerosos sensores. Entre los sensores de los que dispone se cuentan con un sensor de ultrasonidos, un acelerómetro en los 3 ejes y dos cámaras, una frontal y otra vertical, esta última se encuentra situada en la panza del aparato y no dispone de calidad HD como la frontal.

La característica que más le hace destacar por encima de sus competidores en el mercado es que no requiere de un mando físico de control a distancia para su manejo, si no que el propio Ardrone crea una red Wi-Fi a través de la cual nos podemos conectar un dispositivo móvil personal que corra los sistemas operativos de Apple o Google. A través de una aplicación oficial creada para estas plataformas podemos controlar el Ardrone, recibir sus datos de telemetría e incluso una recepción de imágenes en directo.

Ilustración 14: Sistema de comunicación Ardrone comercial

Debido a que Parrot facilitó en código abierto el sistema de control de este producto, nacieron multitud de aplicaciones no oficiales para el control y manejo del cuadricóptero para otros sistemas operativos móviles como Symbian y Linux.

En la siguiente figura obtenida de las Especificaciones del Ardrone 2.0 [16] podemos observar la distribución y características de los diferentes sensores y actuadores incorporados en el dispositivo.

Ilustración 15: Esquema de características de Ardrone 2.0

1. Cámara HD. 720p 30 fps.
2. Objetivo gran angular: 92°, diagonal. Perfil básico de codificación H264. Difusión en tiempo real de baja latencia.
3. Placa Madre
    - Procesador AR M Cortex A8 de 32 bits a 1 GHz con DSP de video TMS320DMC64x a 800 MHz. Linux 2.6.32RAM DDR2 de 1 GB a 200 MHz.
    - USB 2.0 de alta velocidad para extensiones y grabación.
    - Wi-Fi b/g/n.
    - Giroscopio de 3 ejes 2.0000/s.
    - Acelerómetro de 3 ejes +/-50 mg.
    - Magnetómetro de 3 ejes, precisión 6°.
    - Sensor de presión +/- 10 Pa.
    - Sensor de ultrasonidos para medición de altitud respecto al suelo.
    - Cámara QVGA vertical a 60 fps.
4. Tubos de fibra de carbono: peso total de 380 g con el casco de protección para el exterior, 420 g con el casco de protección para el interior.
5. Piezas de plástico nylon cargado con 30% de fibra de vidrio de alta calidad.
6. Espuma para aislar el centro de inercia de las vibraciones de los motores.
7. Casco de protección de polipropileno expandido (PPE) inyectado por un molde metálico.
8. Nanorevestimiento repelente a los líquidos en los sensores de ultrasonidos.
9. 4 motores de rotor interno (“inrunner”) sin escobillas. 14,5 W.

10. Rodamiento de bolas en miniatura.

11. Engranajes de Nylatron de bajo ruido para reductor de hélice de 1/8,75.

12. Eje de las hélices de acero templado.

13. Cojinete de bronce autolubricante.

14. Hélices de alta fuerza de propulsión para mayor maniobrabilidad.

15. Microcontrolador AVR de 8 MIPS.

16. Batería recargable Li-Po de 3 elementos, 1.000 mAh.

17. Conexión USB.

3.1.1 Modificaciones añadidas

A este hardware comercial se le han añadido dos modificaciones artesanales con el objetivo de mejorar ciertos aspectos.

En primer lugar se ha añadido un sistema de amortiguación mediante bridas que previene de forma eficaz roturas de patas y de la cruz central, ante caídas o aterrizajes bruscos. El sistema de amortiguación original deja mucho que desear ya que simplemente se trata de cuatro patas fabricadas en plástico que transmiten todos los esfuerzos recibidos a la cruz central, uno de los elementos más delicados del aparato.

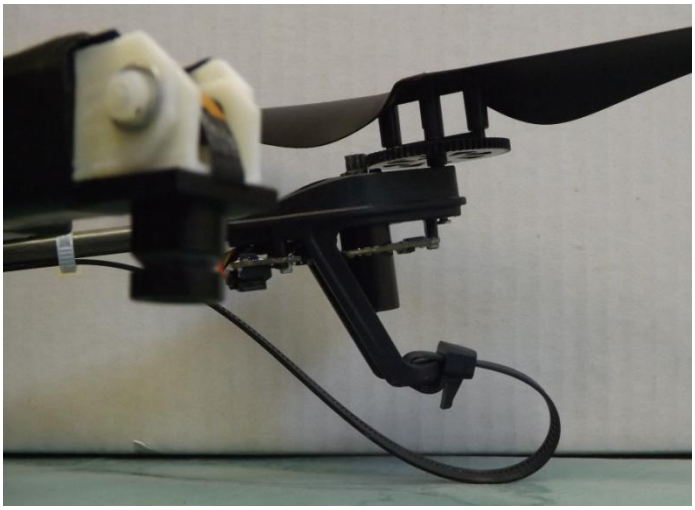

Ilustración 16: sistema de amortiguación mediante bridas

También se le ha realizado una modificación en la cámara frontal, permitiendo que se pueda cambiar su orientación 90º para apuntar hacia el suelo. De esta manera solventamos uno de los grandes problemas que presenta este proyecto, ya que, para realizar la detección de la señal de aterrizaje, la cámara VGA a 60 fps que lleva instalada en la panza, presenta muchos problemas debido a la calidad de la imagen obtenida. Al poder utilizar la cámara frontal HD 720p a 30 fps, la calidad de la detección mejora significativamente, lo cual facilita ampliamente la maniobra de aterrizaje automático.

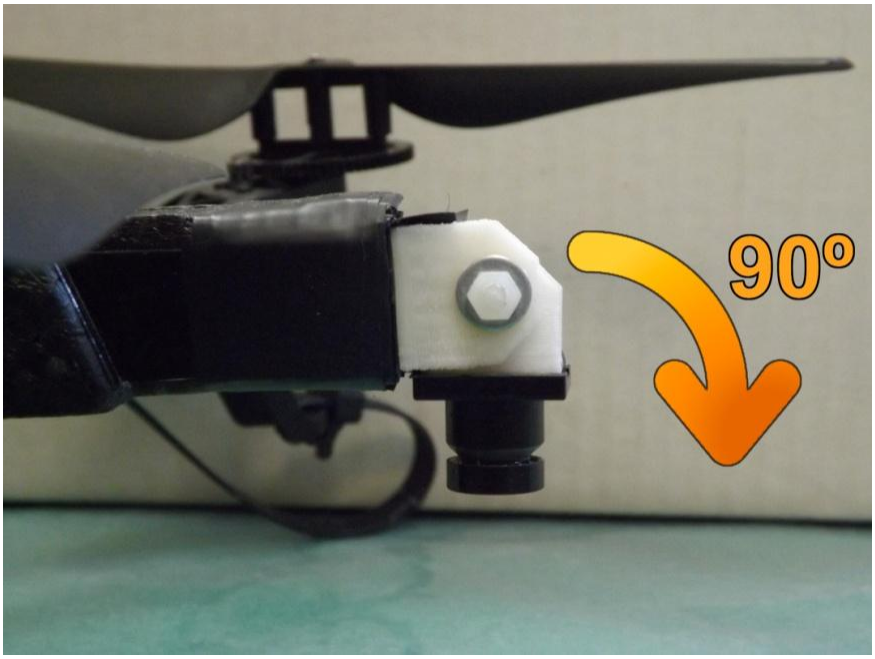

Ilustración 17: Modificación de la cámara frontal

3.2 Software

El Laboratorio de Sistemas Inteligentes de la UC3M está desarrollando un software de control (llamado LSI Drone Interface) para el Ardrone con diversas características de navegación autónoma basado en el SDK (Software development kit) que distribuye abiertamente Parrot para desarrolladores externos a la compañía. Este SDK contiene las librerías necesarias para poder realizar la conexión con el drone, la lectura de los datos de sus sensores y el envío de comandos a los actuadores.

A partir de estas librerías y utilizando el entorno de desarrollo integrado Microsoft Visual Studio y el lenguaje de programación C#, se ha desarrollado una aplicación de Windows muy completa que permite múltiples funcionalidades. Para la parte de visión por computador, se han utilizado las librerías de EMGU CV, que es una plataforma que permite el uso de las librerías de OPEN CV en lenguajes de tipo .NET como son C#, VB, VC++, IronPython etc.

3.2.1 Lenguaje C#

C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos, que permite desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C.

La sintaxis de C# es muy fácil de aprender y utilizar, a la par que muy expresiva. Se basa en signos de llave, por lo que es fácilmente reconocible por cualquier persona familiarizada con C, C++ o Java. El lenguaje C# simplifica muchas de las complejidades de C++ y proporciona características eficaces como tipos de valor que admiten valores NULL, enumeraciones, delegados, expresiones lambda y acceso directo a memoria, que no se encuentran en Java. C# admite métodos y tipos genéricos, que proporcionan mayor rendimiento y seguridad de tipos, e iteradores, que permiten a los implementadores de clases de colección definir comportamientos de iteración

personalizados que el código cliente puede utilizar fácilmente. Todas estas características convierten C# en un lenguaje de primera clase.

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método Main, que es el punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que reemplazan a los métodos virtuales en una clase primaria requieren la palabra clave *override* como medio para evitar redefiniciones accidentales. En C#, una struct es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces pero que no admite la herencia.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen las siguientes:

- Firmas de métodos encapsulados denominadas delegados, que habilitan notificaciones de eventos con seguridad de tipos.
- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- Comentarios en línea de documentación XML.
- Language-Integrated Query (LINQ) que proporciona funciones de consulta integradas en una gran variedad de orígenes de datos.

Esta información ha sido extraída de la web oficial de Microsoft C# [17].

3.2.2 Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado de Microsoft. Se utiliza para desarrollar programas informáticos para la familia de los sistemas operativos de Microsoft Windows, así como sitios web, aplicaciones web y servicios web. Visual Studio utiliza plataformas de desarrollo de software de Microsoft, como la API de Windows, Windows Forms, Windows Presentation Foundation, Windows Store y Microsoft Silverlight. Puede producir tanto código nativo como código administrado.

Visual Studio incluye un editor de código con apoyo IntelliSense. El depurador integrado funciona tanto como un depurador a nivel de fuente y un depurador a nivel de máquina. Otras herramientas integradas incluyen un diseñador de formularios para crear aplicaciones GUI, diseñador web, diseñador de la clase, y bases de datos con diseño de esquema. Acepta plug-ins que mejoran la funcionalidad en casi todos los niveles, incluyendo la adición de soporte para los sistemas de control de código fuente y la adición de nuevos conjuntos de herramientas como editores y diseñadores visuales para idiomas o de dominio específico para otros aspectos del ciclo de vida de desarrollo de software (como el cliente de Team Foundation Server: Team Explorer).

Visual Studio es compatible con diferentes lenguajes de programación y permite al editor de código y depurador apoyar (en diversos grados) casi cualquier lenguaje de programación, siempre que exista un servicio específico del lenguaje. Los lenguajes que incluye son C, C++ y C++/CLI (a través de Visual C++), VB.NET (a través de Visual Basic. NET), C# (mediante Visual C#) y F# (a partir de Visual Studio 2010). Incluye soporte para otros idiomas como M, Python y Ruby, entre otros que se encuentran disponibles a través de los servicios de lenguajes instalados por separado.

Microsoft ofrece ediciones "Express" de su Visual Studio sin costo alguno. Las versiones comerciales de Visual Studio, junto con ciertas versiones anteriores están disponibles de forma gratuita a los estudiantes a través del programa DreamSpark de Microsoft [18].

Información obtenida de la página oficial de Microsoft Visual Studio [19].

3.2.3 SDK

El SDK o Kit de Desarrollo de Software permite a desarrolladores externos a la compañía crear y distribuir nuevas aplicaciones o juegos basados en el Ardrone 2.0. Para descargar estas librerías de desarrollo en necesario registrarse en la página web del fabricante y aceptar una serie acuerdos y condiciones.

Este SDK incluye:

- Un documento explicativo de cómo usar el SDK, donde también se describen los protocolos de comunicación del drone.
- La librería AR.Drone 2.0 (ARDroneLIB), la cual proporciona las APIs necesarias para una fácil comunicación y configuración del drone.
- La librería de herramientas (ARDroneTool), que ofrece un cliente del drone donde los desarrolladores sólo tienen que insertar su código específico de aplicaciones personalizadas;
- La librería de control de motores.
- El código fuente para iOS y Android de la aplicación AR.FreeFlight 2.0.

3.2.4 Emgu CV

Emgu CV son unas librerías multiplataforma permite el uso de las librerías de OPEN CV en lenguajes de tipo .NET como son C#, VB, VC++, IronPython etc.

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel, que es utilizada en infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento a aplicaciones de control de procesos industriales donde se requiere el reconocimiento de objetos.

Emgu CV está escrito por completo en C#. Tiene la ventaja de que se puede compilar en Mono y por lo tanto es capaz de funcionar en cualquier plataforma con soporte Mono, incluyendo Linux, Mac OS X, iOS y Android. Ha requerido una gran cantidad de de esfuerzos por parte de sus creadores para poder tener una aplicación en C# puro, ya que, los encabezados tienen que ser adaptados, en comparación con C++, donde los

archivos de cabecera simplemente pueden ser incluidos. Emgu CV se puede utilizar en varios lenguajes diferentes, incluyendo C #, VB.NET, C + + y IronPython.

En cuanto a la arquitectura, Emgu CV tiene dos capas como se muestra a continuación:

- La capa de base (layer 1) contiene funciones, estructuras y enumeración de asignaciones que son reflejo directo de las OPEN CV.
- La segunda capa (layer 2) contiene clases que incorporan las ventajas que supone el mundo .NET.

Esta información ha sido obtenida de la web oficial de Emgu CV [20].

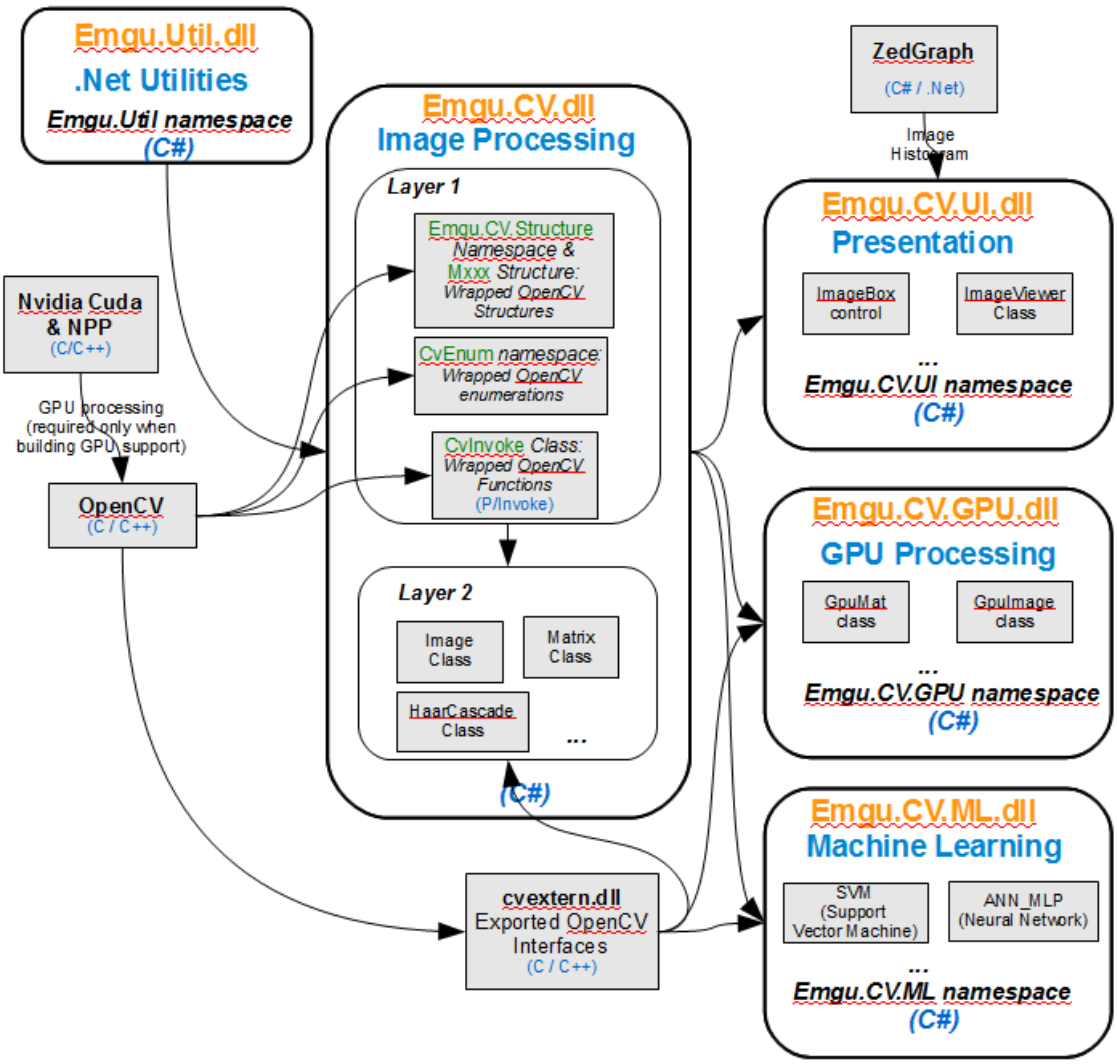

Ilustración 18: Esquema de arquitectura de Emgu CV

3.2.5 LSI Drone Interface

Integrantes del Laboratorio de Sistemas Inteligentes de la UC3M está desarrollando una aplicación con interfaz grafica para el control y vuelo autónomo del Ardrone basado utilizando las librerías facilitadas en el SDK de Parrot y usando características de visión por computador apoyadas en las librerías de Emgu CV.

El software dispone de múltiples características como se muestran a continuación:

Ilustración 19: LSI Drone Interface

Esta aplicación todavía se encuentra en fase de desarrollo, por lo que algunas características no tienen aun un funcionamiento optimo. La parte que más nos afecta para la realización de este trabajo es el sistema de control y estabilidad. Actualmente, solo se dispone de un control de tipo proporcional que permite al aparato mantenerse en el aire, pero no de una forma suficientemente estable. Se producen desplazamientos laterales y diversos cambios bruscos de altura que afectan a la estabilidad del sistema y en caso de este trabajo, a la imagen obtenida por la cámara.

Capítulo 4: COMPARATIVA SIFT-SURF

Buscar correspondencias entre dos imágenes de la misma escena o que contengan el mismo objeto forma parte de muchas aplicaciones de visión por computador. Por eso ha sido un campo muy activo en el entorno de la investigación de este tipo de tecnologías en los últimos años.

En 1999 Lowe presenta SIFT (abreviatura) para la extracción de características distintivas desde imágenes que pueden ser invariantes a escala y rotación. A partir de ese momento esto fue usado para multitud de aplicaciones como crear mosaicos de imágenes, reconocimiento, recuperación, etc. En 2006, Bay y Tuytelaars crean SURF, con el objetivo de acelerar el proceso y obtener características más robustas, para ello usan la convolución en las imágenes y detector Fast-Hessian. Este nuevo experimento resultó satisfactorio, ya que, era más rápido y funcionaba bien.

Estos algoritmos tienen en común que se dividen en dos fases principales. En primer lugar, los puntos clave o key points son extraídos de diferentes lugares de la imagen como bordes, manchas, esquinas, etc. Después las regiones vecinas de cada key point son recogidas y a partir de esos datos se calculan los diferentes descriptores característicos.

Durante el proceso, las características de cada imagen son extraídas y almacenadas. Estas características tienen que ser lo sufrientemente descriptivas a la par que robustas ante el ruido y distintos errores. Por último se establecen las correspondencias entre los descriptores de las diferentes imágenes. La correspondencia de descriptores está basada en distancias, como la distancia euclidiana.

4.1 Algoritmo SIFT

El algoritmo SIFT fue propuesto por Lowe en 1999 [21] para resolver el problema que existía con rotaciones, escalados, deformaciones afines, cambios del punto de vista, ruido, cambios en la iluminación, etc. en la imagen.

Este algoritmo tiene cuatro partes bien diferenciadas:

1. Detección de los extremos de escala-espacio
2. Localización de los key points
3. Asignación de orientaciones
4. Generación del descriptor

La primera etapa consta en aplicar la diferencia gausiana (DoG, difference of gaussian) con diferentes tamaños de región y buscar máximos locales tanto en el espacio de la imagen como de la escala.

Para una escala concreta, la DoG ofrece una respuesta muy fuerte para esquinas de tal tamaño que encajan con dicha escala. Por tanto, en esta etapa comparamos cada punto de la escala-espacio con los vecinos de esa misma escala y de las escalas anteriores y posteriores.

Después de encontrar los key points, se depuran con precisión sub-pixel usando la expansión de la serie de Taylor de la escala-espacio. Si el valor del punto extremo encontrado es inferior al valor establecido en por cierto umbral, el punto es automáticamente descartado.

La DoG ofrece picos de valor al analizar los bordes, por lo que estos deben ser eliminados para el correcto funcionamiento del algoritmo. Con este fin se hace uso de la matriz hessiana, que calcula las curvaturas principales, para quedarse solo con unos valores que no difieran en exceso de un orden de magnitud establecido, y evitar fallos.

Seguidamente se le asigna una orientación a cada key point de manera que se garantiza la invariancia respecto a la rotación de las imágenes. Con este fin se observan los puntos vecinos alrededor de cada key point (en función de la escala) y se computa la dirección del gradiente y la magnitud. Después se crea un histograma de dichas direcciones ponderado por la magnitud del gradiente. El máximo que observamos en el histograma muestra la orientación del key point. Si existen otros máximos superiores al 80% del principal, son usados para definir otros key points en la misma escala y posición pero con distinta orientación.

Posteriormente se forman los descriptores de los key points. Para cada punto se toma un vecindario de 16x16 puntos. Este, a su vez, se fracciona en sub-bloques de dimensión 4x4 y para cada uno de ellos se da forma a un nuevo histograma de orientaciones. El encadenamiento en un vector de valores de las cajas de cada histograma para los 16 sub-bloques obtenidos del key point compone su descriptor.

La correspondencia entre los key points de dos imágenes se consigue a través de una exploración del punto más cercano en el espacio de los descriptores de key points. Ocurre en ocasiones que el segundo punto inmediato puede estar muy próximo del primero a causa del ruido. Por ello se computa la razón entre la distancia al más cercano y al siguiente más cercano y si el resultado obtenido se encuentra por encima de cierto umbral previamente definido, los puntos son descartados.

4.2 Algoritmo SURF

El algoritmo SURF es otro de los más utilizados en el área del tratamiento digital de la imagen para la extracción de key points invariantes, presentado en 2006 por Bay y Tuytelaars [22]. La extracción de los puntos la efectúa en primer lugar detectando los posibles key points y su situación dentro de la imagen. A continuación se representa la vecindad del key point como un vector de características (que lleva preestablecido un tamaño de 64, siendo posible realizar una modificación para aumentar su dimensión).

Este algoritmo maneja una aproximación simple de la matriz Hessiana para reducir el tiempo de computación.

Esta matriz Hessiana es empleada debido a su excelente proporción entre el coste computacional y la precisión que aporta. El determinante de la Hessiana es utilizado para la situación de los puntos y para la determinación de la escala.

El descriptor SURF tiene algún parecido al planteado en SIFT. El primer paso para definir el descriptor teniendo ya supuesta la escala es el cálculo de la orientación del key point para, posteriormente, calcular el descriptor SURF.

Con el fin de obtener un punto que no varié frente a cambios en la orientación se calcula el "Haar-wavelet" para las direcciones *x* e *y* en una región circular de radio *6s*, donde *s* es la escala del key point.

Cuando ya los tiene calculados para todos los vecinos, se aproxima la orientación predominante computando la suma de todos los resultados obtenidos dentro de una ventana que se desliza cubriendo un ángulo de $\frac{\pi}{3}$. El cálculo del descriptor se efectúa construyendo, en primer lugar, una región cuadrada centrada en el key point con un tamaño de *20s*. Esta región es dividida de forma regular en 4 sub-regiones, y para cada sub-región se examinan algunas características sencillas. Inmediatamente después se calculan la "Haar-wavelet" para *x* e *y* y se pulen los resultados obtenidos a través de una Gaussiana, obteniendo *dx* y *dy*. Para cada sub-región se computa una suma de los resultados *dx* y *dy*, asimismo se calcula su valor absoluto $|dx|$ y $|dy|$. De este modo, de cada sub-región se obtiene un vector *v* compuesto por $v = (\sum dx, \sum dy, \sum |dx|, \sum |dy|)$. El descriptor SURF es obtenido a través de la unión de los vectores de las sub-regiones.

La principal característica de los key points de SURF es la repetibilidad, si se considera fiable un punto, el detector hallará el mismo punto desde distintos puntos de vista (diferente escala, orientación, etc). El vector obtenido por el algoritmo debe ser característico a la par que robusto al ruido, errores, y deformaciones geométricas y fotométricas.

4.3 Estudio comparativo SIFT-SURF

Como ya hemos visto ambos métodos son usados para obtener un vector descriptor de cada key point, aunque lo obtienen de distinta manera. Por lo que ambos métodos serian validos para la aplicación que queremos realizar.

Para realizar estas comparaciones usaremos las funciones definidas en Emgu CV de *SIFTDetector()*, que es un detector SIFT estándar; y las funciones *SURFDetector(1000, false)* y *SURFDetector(500, false)*, en este caso se trata de un detector SURF donde el termino numérico indica el *hessianThresh* (el valor del hessiano al partir del cual los key point son aceptados) y el termino booleano *false* (que indica que se trata de un descriptor básico de 64 bits). Se han elegido estos valores para intentar reducir al máximo el coste computacional, ya que para la aplicación que vamos a realizar no interesa usar el algoritmo que nos ofrezca la mejor relación calidad/tiempo.

Para realizar la comparación vamos a utilizar estos algoritmos para obtener los key points de una serie de imágenes. Las imágenes estarán divididas en cuatro grupos: Imágenes aéreas, de personas, de edificios y de calles. En total se examinaran más de 4000 imágenes y se anotara tanto el número de descriptores obtenidos como el tiempo que se ha empleado en la operación. Además, calcularemos la relación $\frac{descriptores}{tiempo}$ que nos dará la información que estamos buscando.

Las pruebas se realizaran en un ordenador con procesador *Pentium Dual-Core E5400 @ 2.70GHz, 4.00 GB de RAM* y un sistema operativo *Windows 8.1 Pro x64* a través de un programa creado en Visual Studio en lenguaje C#.

A continuación se expone una muestra de las imágenes utilizadas en el experimento, así como las tablas con las estadísticas obtenidas.

Ilustración 20: Muestra de imágenes de edificios

Ilustración 21: Muestra de imágenes de personas

Ilustración 22: Muestra de imágenes aéreas

Ilustración 23: Muestra de imágenes de calles

Tabla 1: Estadísticas de imágenes de edificios

Imágenes de edificios									
Estadística	SIFT Keypoints	SIFT time	SIFT Points/ms	SURF500 Keypoints	SURF500 time	SURF500 Points/ms	SURF1000 Keypoints	SURF1000 time	SURF1000 Points/ms
No. de observaciones	1659	1659	1659	1659	1659	1659	1659	1659	1659
Mínimo	0,000	70,000	0,000	0,000	111,000	0,000	0,000	93,000	0,000
Máximo	8068,000	917,000	8,798	11881,000	3005,000	4,115	8381,000	2542,000	4,185
1° Cuartil	737,500	239,500	3,055	1304,500	701,000	1,831	755,000	527,000	1,399
Mediana	1123,000	291,000	3,869	2154,000	912,000	2,370	1283,000	678,000	1,917
3° Cuartil	1537,000	348,000	4,464	3244,000	1186,000	2,772	2101,000	896,000	2,386
Media	1195,383	299,970	3,730	2396,090	966,173	2,287	1538,683	731,659	1,896
Varianza (n-1)	414320,957	7399,693	1,060	2089312,534	132943,162	0,449	1172726,103	81220,516	0,490
Desviación típica	643,678	86,021	1,030	1445,445	364,614	0,670	1082,925	284,992	0,700

Tabla 2: Estadísticas de imágenes de personas

Imágenes de personas									
Estadística	SIFT Keypoints	SIFT time	SIFT Points/ms	SURF500 Keypoints	SURF500 time	SURF500 Points/ms	SURF1000 Keypoints	SURF1000 time	SURF1000 Points/ms
No. de observaciones	901	901	901	901	901	901	901	901	901
Mínimo	19,000	9,000	1,024	34,000	15,000	0,642	21,000	13,000	0,370
Máximo	4272,000	682,000	6,289	6890,000	1473,000	5,854	4962,000	1137,000	5,094
1° Cuartil	403,000	105,000	3,705	669,000	221,000	2,894	403,000	175,000	2,200
Mediana	555,000	126,000	4,295	1038,000	286,000	3,482	664,000	226,000	2,795
3° Cuartil	807,000	189,000	4,724	1516,000	415,000	4,051	971,000	333,000	3,371
Media	682,559	158,526	4,190	1225,353	352,210	3,415	786,380	278,569	2,752
Varianza (n-1)	211349,089	8112,623	0,714	707819,218	44688,499	0,699	326441,951	27537,252	0,717
Desviación típica	459,727	90,070	0,845	841,320	211,397	0,836	571,351	165,944	0,847

Tabla 3: Estadísticas de imágenes aéreas

Imágenes aéreas									
Estadística	SIFT Keypoints	SIFT time	SIFT Points/ms	SURF500 Keypoints	SURF500 time	SURF500 Points/ms	SURF1000 Keypoints	SURF1000 time	SURF1000 Points/ms
No. de observaciones	582	582	582	582	582	582	582	582	582
Mínimo	23,000	37,000	0,545	18,000	60,000	0,269	0,000	50,000	0,000
Máximo	17516,000	3081,000	6,811	42795,000	8997,000	5,271	27196,000	6122,000	4,814
1° Cuartil	1404,750	331,000	4,254	2925,000	812,000	3,699	1672,500	593,250	3,079
Mediana	1932,500	403,000	4,962	4239,500	1043,000	4,086	2956,000	813,000	3,582
3° Cuartil	2483,500	468,750	5,407	6109,250	1353,500	4,470	4276,500	1017,750	4,094
Media	1946,979	390,296	4,667	4340,552	1031,849	3,832	3040,009	802,655	3,353
Varianza (n-1)	1150631,66	29903,30	1,293	7501560,978	292928,98	1,098	4098729,699	168986,233	1,292
Desviación típica	1072,675	172,926	1,137	2738,898	541,229	1,048	2024,532	411,079	1,137

Tabla 4: Estadísticas de imágenes de calles

Imágenes de calles									
Estadística	SIFT Keypoints	SIFT time	SIFT Points/ms	SURF500 Keypoints	SURF500 time	SURF500 Points/ms	SURF1000 Keypoints	SURF1000 time	SURF1000 Points/ms
No. de observaciones	868	868	868	868	868	868	868	868	868
Mínimo	593,000	297,000	1,977	1079,000	609,000	1,746	678,000	526,000	1,209
Máximo	3810,000	683,000	5,755	8554,000	1857,000	4,705	5900,000	1395,000	4,254
1° Cuartil	1417,500	399,000	3,546	3516,750	1048,250	3,371	2162,750	811,750	2,664
Mediana	1728,500	435,500	3,966	4370,500	1180,500	3,686	2776,000	913,000	3,058
3° Cuartil	2061,000	477,000	4,339	5231,250	1322,250	3,960	3405,000	1024,250	3,367
Media	1756,631	438,750	3,924	4375,552	1185,432	3,616	2821,988	917,722	2,993
Varianza (n-1)	247745,506	3586,801	0,362	1613696,564	44371,669	0,236	864761,737	25137,322	0,283
Desviación típica	497,740	59,890	0,601	1270,314	210,646	0,486	929,926	158,548	0,532

4.4 Conclusiones

Como podemos observar en las tablas estadísticas obtenidas y en los experimentos realizados el algoritmo *SURFDetector(1000, false)* queda totalmente descartado, ya que, es el que menos key points ofrece además de presentar el peor ratio de puntos/ms. Se ha comprobado experimentalmente que no se consigue realizar una detección adecuada del patrón en las condiciones normales de vuelo del drone.

Comparando *SIFTDetector()* y *SURFDetector(500, false)* podemos observar que el detector SURF nos ofrece aproximadamente el doble de key points que el SIFT, a costa de emplear más del doble de tiempo. El SIFT presenta un ratio de puntos/ms superior al SURF en todos los casos examinados, realizando una detección de key points más eficientes.

Se ha podido comprobar experimentalmente que el algoritmo *SURFDetector(500, false)* realiza una detección del patrón más precisa, pero a costa de un coste computacional que no es asumible, ya que al tratarse de una aplicación que tiene un funcionamiento en tiempo real, debemos de tratar de minimizar al máximo los tiempo del procesamiento de imágenes.

Por lo tanto usaremos el *SIFTDetector()*, puesto que, nos ofrece la mejor relación de puntos/ms además de una cantidad suficiente de key points para realizar una detección correcta del patrón en un tiempo que podemos considerar aceptable para la realización de este trabajo.

Capítulo 5: DISEÑO DEL SISTEMA

5.1 Aplicación independiente

El primer paso para realizar este trabajo se basa en la creación de una aplicación con la que se pueda experimentar como se realiza la detección sin tener que involucrar al drone, con las complicaciones y los peligros que ello supone.

Para ello se crea una aplicación que permite obtener una imagen desde cualquier cámara conectada al ordenador y analizar su contenido. Mediante un algoritmo SIFT se extraen las características tanto de la imagen objeto como de la imagen observada para después realizar el emparejamiento de características y obtener la matriz de homografía. Esta matriz de homografía nos permitirá situar los puntos sobre la imagen observada, obteniendo nuestra ROI. Además, esta aplicación también nos muestra por pantalla las coordenadas relativas del centro de la ROI con respecto al centro de la imagen obtenida por la cámara, el ángulo de rotación de esta y el tiempo que se ha empleado en todo el proceso de detección de la imagen.

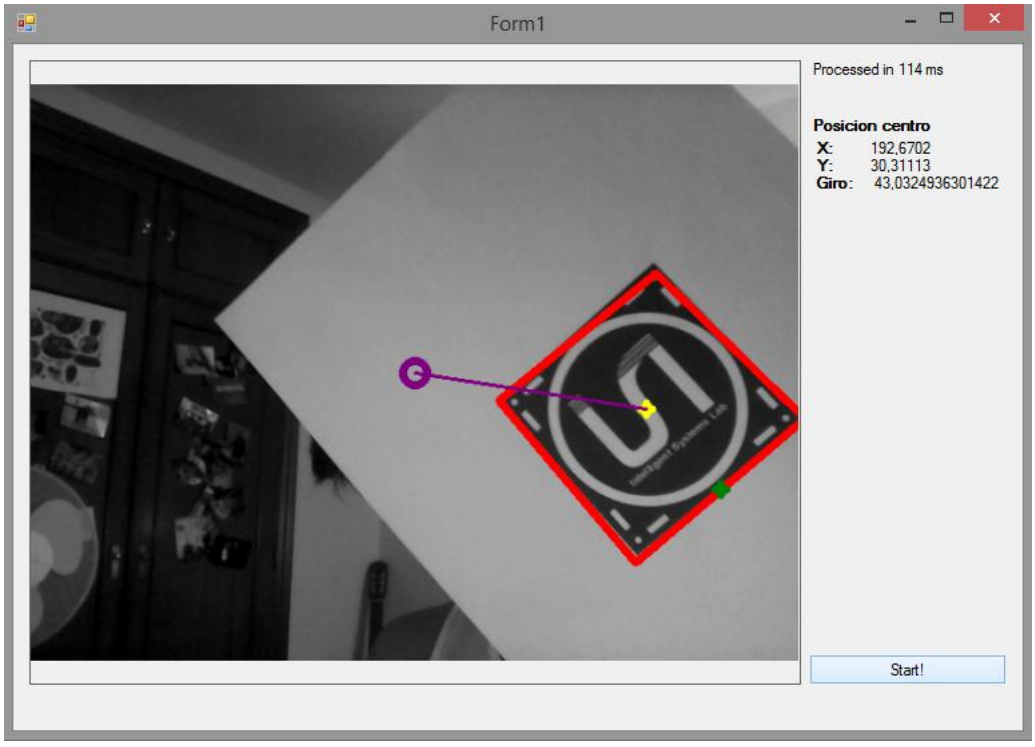

Ilustración 24: Detección en aplicación independiente

A partir de los puntos obtenidos mediante la detección del patrón obtenemos las coordenadas relativas del centro de la ROI con respecto al centro de la imagen obtenida por la cámara de la siguiente manera:

$$pos\_x = center\_ROI.X - center\_CAM.X$$

$$pos\_y = center\_ROI.Y - center\_CAM.Y$$

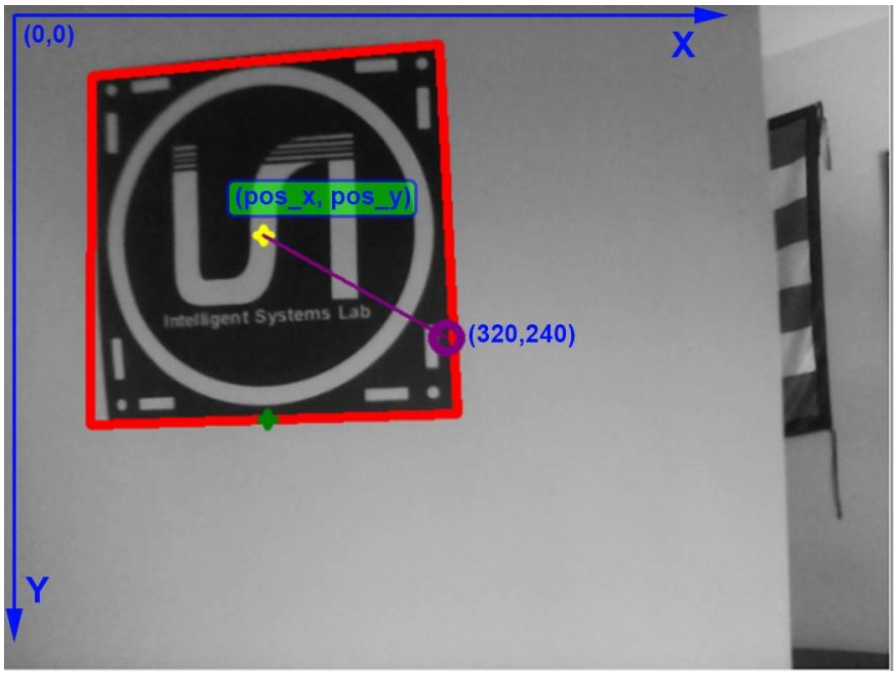

Ilustración 25: Obtención de coordenadas (x,y)

Y el ángulo de rotación del patrón sobre la imagen obtenida lo obtenemos a través de la siguiente fórmula aplicada como se puede apreciar en la ilustración 26.

$$beta\_angle = \frac{180}{\pi} \arctan \frac{center\_ROI.X - bottom\_ROI.X}{center\_ROI.Y - bottom\_ROI.Y}$$

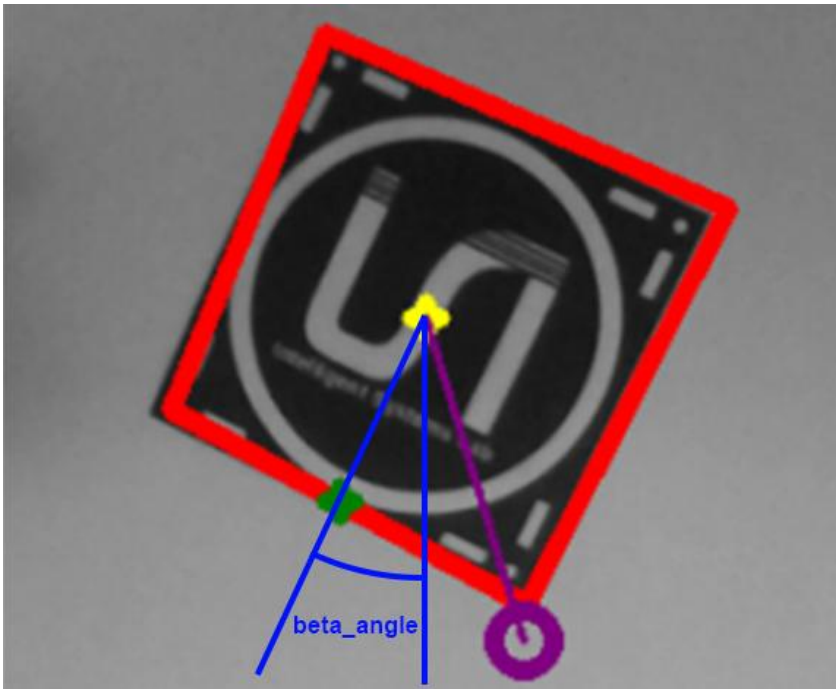

Ilustración 26: Obtención del ángulo de rotación

En los siguientes diagramas de flujo podemos observar el funcionamiento de esta aplicación. En la ilustración 27 se muestra el diagrama de la aplicación general, mientras que en la ilustración 28 se muestra el funcionamiento de la función DrawMatches, a la que se hace referencia en el programa principal.

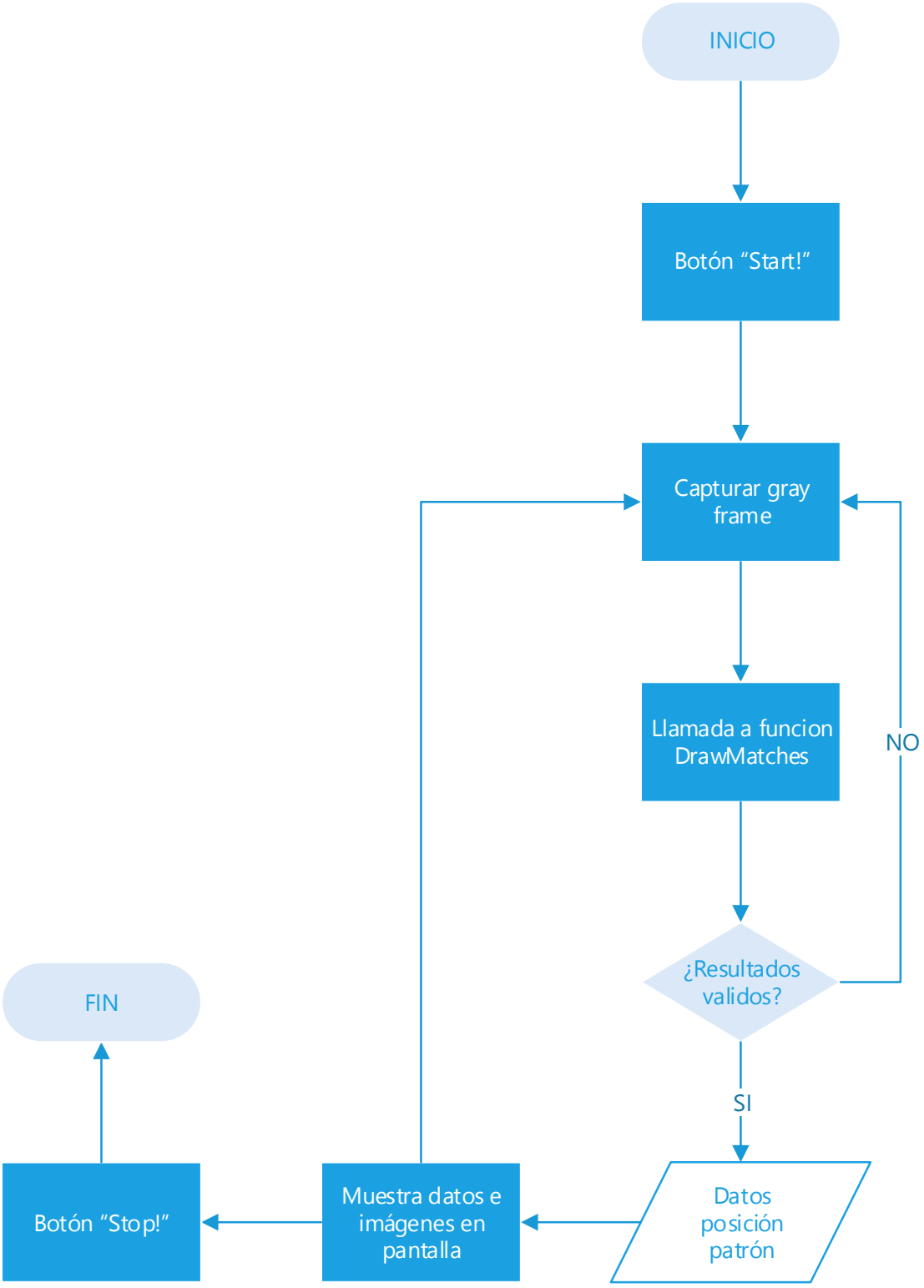

Ilustración 27: Diagrama de la aplicación general

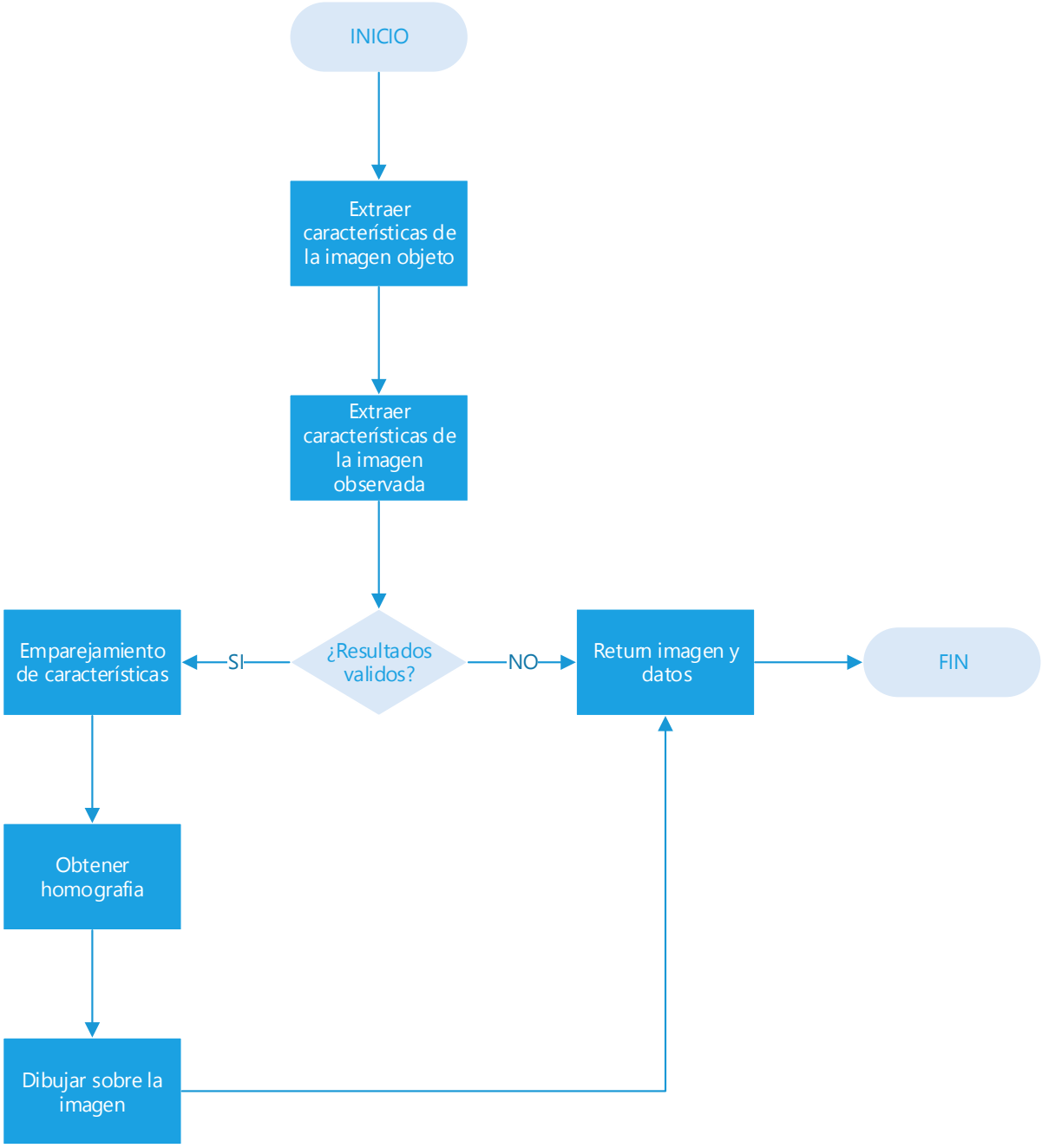

Ilustración 28: Diagrama de la función DrawMatches

5.2 Integración en LSI drone interface

Una vez conseguida una aplicación independiente que nos permite realizar la detección y obtención de los parámetros de posicionamiento del patrón de forma satisfactoria, el siguiente paso consiste en integrar, a partir del código utilizado para la primera aplicación creada, estas funcionalidades en la LSI drone interface. Posteriormente se utilizaran los datos obtenidos a través del reconocimiento del patrón para analizarlos y transmitirlos al sistema de navegación con el fin de realizar las maniobras que sean requeridas.

Para ello se ha creado un botón que activa la función de aterrizaje automático, "Autoland". Cuando esta función está activa comienza el procesado de las imágenes obtenidas con el fin de reconocer y situar el patrón, también llamado "helipad".

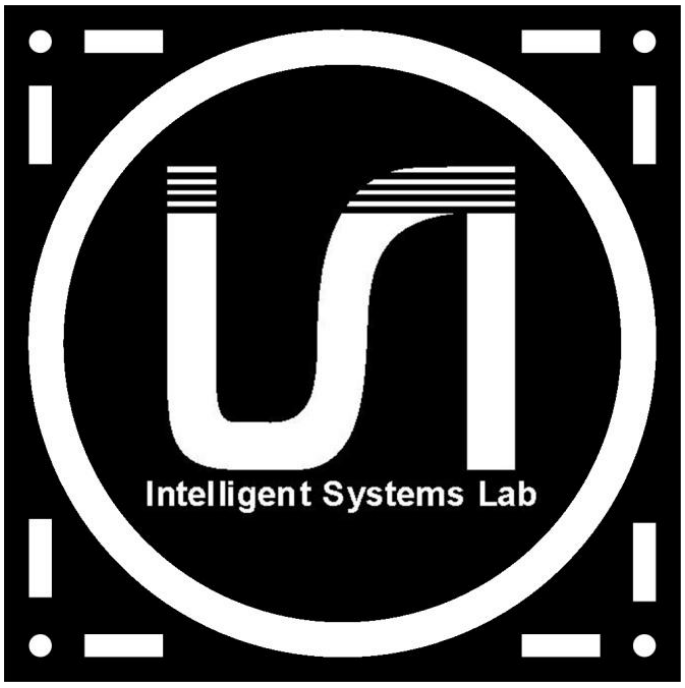

Ilustración 29: Helipad LSI

Si el patrón no es detectado, el drone seguirá con la navegación que llevara establecida antes de activar la función Autoland, pero sí el patrón es detectado se activará el modo de control automático que llevara al drone a situarse justo encima del helipad para realizar el aterrizaje en la zona requerida. Esto se consigue a partir de los datos

obtenidos de la posición relativa del helipad con respecto al centro de la imagen captada por el drone. Estos datos son procesados para poder ser interpretados por el sistema de control, lo cual se consigue a través de un sistema de control en lazo cerrado con un regulador proporcional.

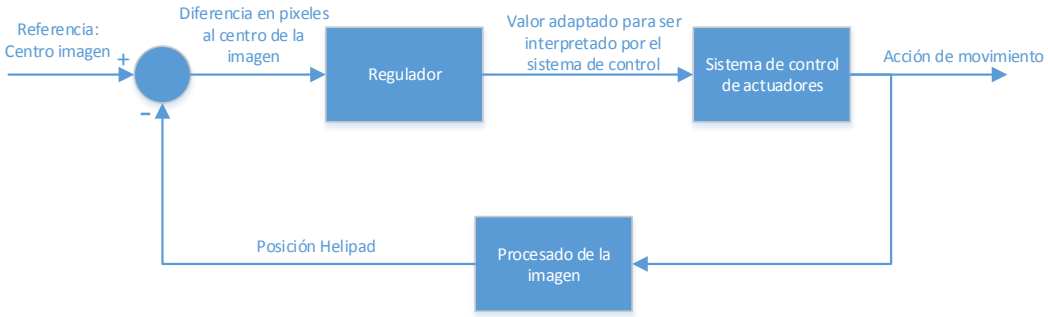

Ilustración 30: Sistema de control en lazo cerrado

A través de los valores obtenidos de la posición del helipad se calcula la diferencia al centro de la imagen. Ese dato numérico de la distancia en pixeles necesita ser transformado para que sea correctamente interpretado por el sistema de control de los actuadores, que modificara la potencia de cada rotor para cambiar los ángulos de roll, pitch y yaw con el objetivo de producir el movimiento deseado en la aeronave.

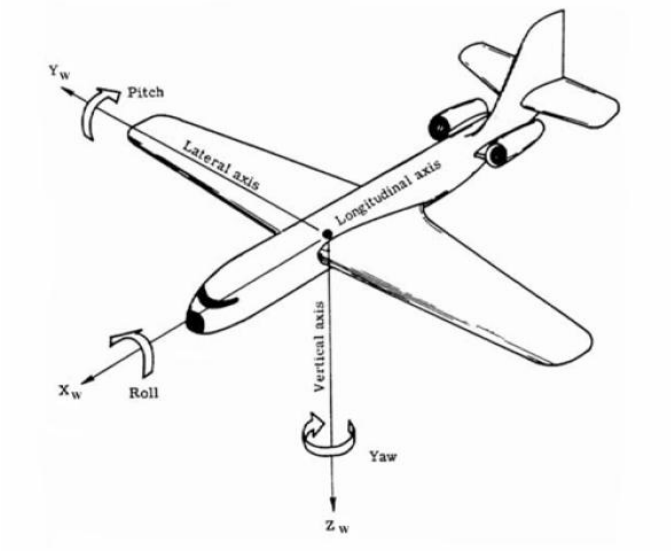

Ilustración 31: Ángulos de una aeronave

Para el regulador se busca ante todo la estabilidad del sistema, ya que, sí introducimos acciones de control demasiado bruscas perderemos firmeza en la obtención de la imagen, lo cual empeora la calidad de la detección. Es por ello que se debe tener mucho cuidado al elegir la ganancia por la que multiplicaremos la distancia en pixeles obtenida a través del reconocimiento del helipad.

En la ilustración 32 podemos observar el diagrama de flujo que aclara la actividad de la función denominada “Autoland”

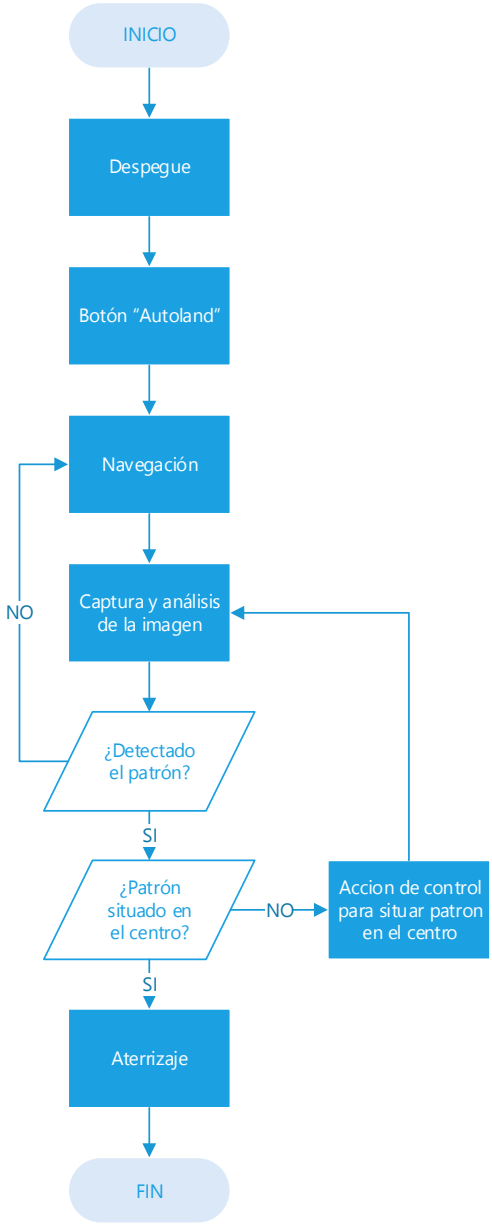

Ilustración 32: Diagrama de Autoland

Una vez detectado el helipad tenemos que analizar si su centro se encuentra dentro la región que definimos como "zona de aterrizaje" o si se encuentra fuera de esta. Como la imagen obtenida por la cámara posee una resolución de 1280 x 720 pixeles, la zona de aterrizaje se define como un área de 300 x 300 pixeles, donde se da por buena la situación del drone sobre el helipad y se procede a lanzar el comando de aterrizaje. Cuanto menor sea el área de esta región de tolerancia más preciso será el aterrizaje justo encima del helipad, a la vez que requerirá de más exactitud en el control y más tiempo empleado para situarse en la posición correcta.

Si el centro del helipad detectado se encuentra fuera de la zona de aterrizaje se pone en marcha el modo de control automático, que hará navegar al drone de tal manera que el centro del helipad se sitúe dentro de la zona de aterrizaje.

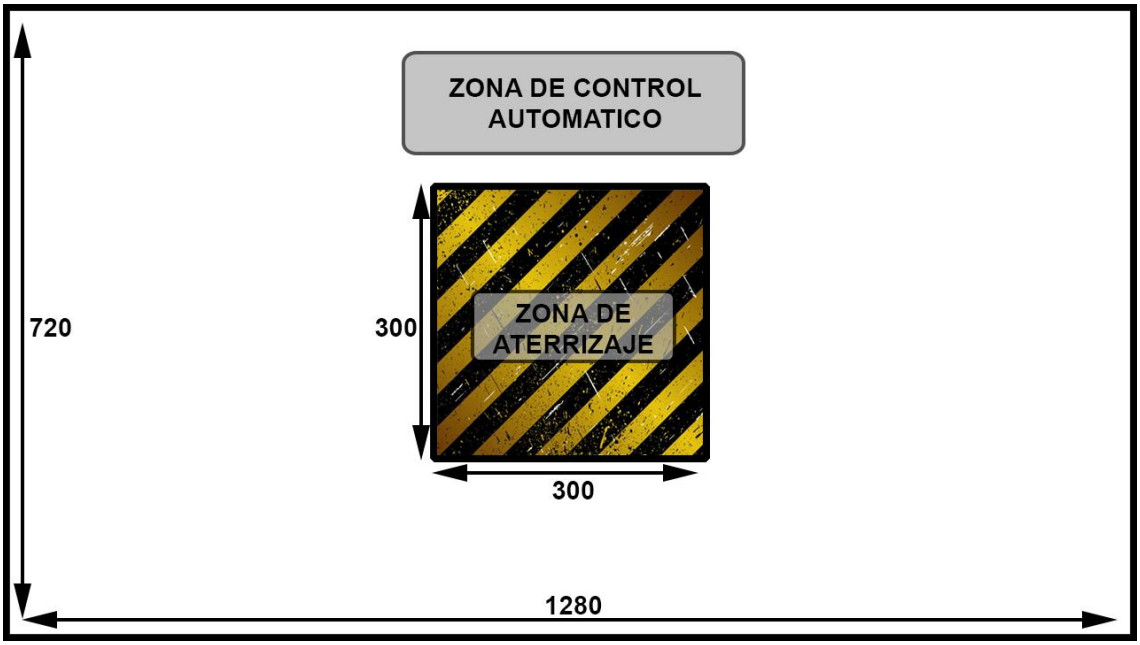

Ilustración 33: Esquema de zonas Autoland

Capítulo 6: RESULTADOS

A partir de un vehículo aéreo no tripulado comercial diseñado con un fin recreativo, el Laboratorio de Sistemas Inteligentes de la UC3M ha creado una aplicación para convertir este vehículo en un sistema aéreo no tripulado (UAS), capaz de realizar navegación de forma autónoma basándose en la información recibida por sus sensores, principalmente a través de las dos cámaras que lleva incorporadas.

Partiendo de esa base se ha diseñado un modulo que de expansión para que el Ardrone, que a través de visión por computador, permite que se realice el acercamiento y aterrizaje sobre una zona identificada con el helipad para tal efecto. Esto se ha diseñado de tal forma que, una vez que esta activada la función de Autoland, el Ardrone pueda estar realizando navegación autónoma o manual hasta el momento que el helipad sea detectado dentro del rango de captación de la cámara, momento en el cual se activara un control de navegación autónomo que situara el Ardrone justo encima de la zona de aterrizaje, antes de lanzar el comando necesario para que la aeronave tome tierra.

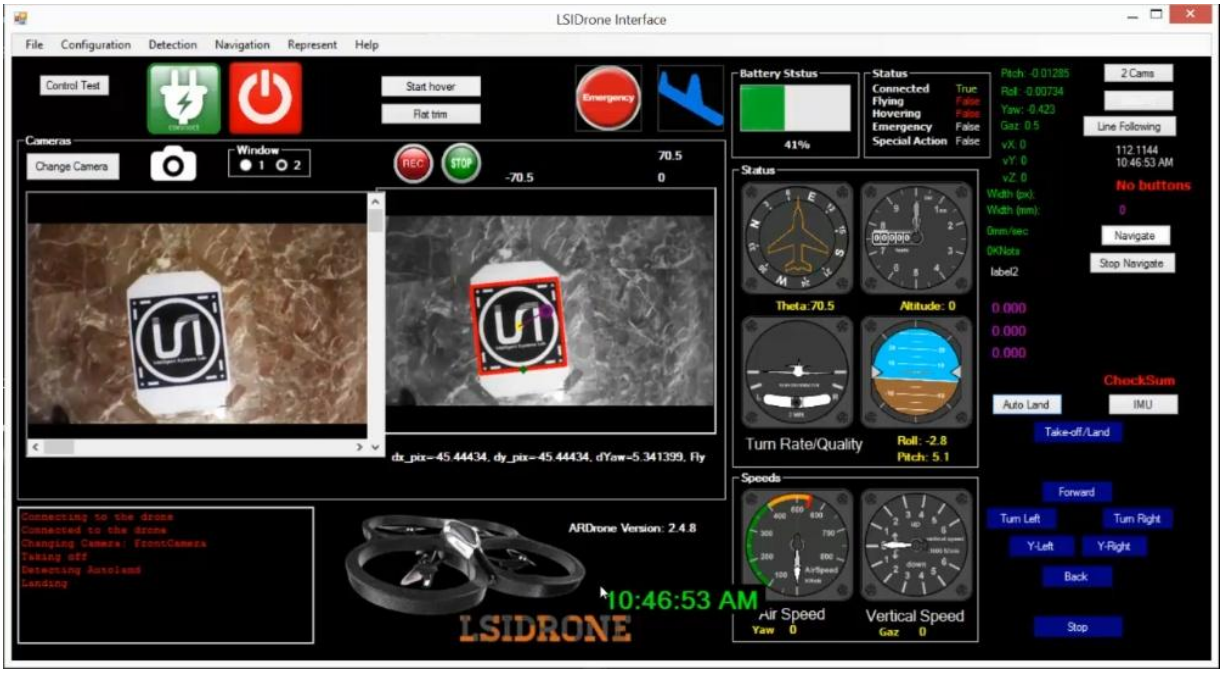

Ilustración 34: LSI Drone Interface Autolanding

Capítulo 7: TRABAJOS FUTUROS Y CONCLUSIONES

7.1 Trabajos futuros

En este proyecto se ha desarrollado la manera de utilizar las imágenes obtenidas por la cámara para posicionar el drone sobre una zona concreta. A través de pequeñas modificaciones se podrían usar los mismos métodos para realizar un seguimiento de un objeto que se encuentra en movimiento, manteniendo el drone posicionado sobre este.

Otra posible expansión de esta aplicación se puede basar en la realizar un despegue y direccionamiento automático con el objetivo de poder seguir una ruta marcada hasta encontrar el helipad y realizar el aterrizaje autónomo, como se puede observa en la ilustración 35. También en estas balizas de seguimiento se podría incluir información visual que ordenara al drone tareas como hacer un cambio de altitud, enviar información a la base acerca de su posición o simplemente tomar una foto.

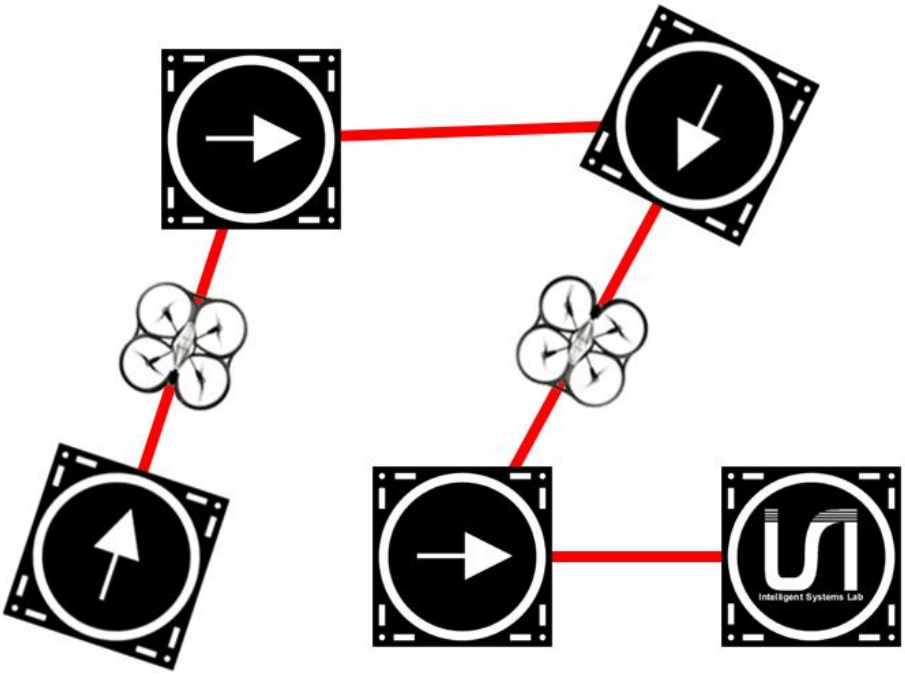

Ilustración 35: Sistema de seguimiento de balizas

7.2 Conclusiones

Vista la capacidad que tiene la visión por computador para apoyar la realización de vuelos o maniobras autónomas en aeronaves no tripuladas, se puede sacar la conclusión de que los drones con cuatro rotores en configuración de cuadricóptero ofrecen una plataforma extraordinaria en cuanto a estabilidad, maniobrabilidad y sencillez para la captación y análisis de imágenes.

Estas características permiten que la investigación basada este campo crezca de forma exponencial en los últimos años, de manera que cada vez se encuentran más aplicaciones tanto de uso civil como militar. Ya existen multitud de vehículos no tripulados, como vimos en el apartado "2.1 Tecnología UAV", pero casi ninguno de los que están en el mercado cuentan con la capacidad de realizar vuelos y búsquedas de forma autónoma. Una aplicación muy interesante seria usar vehículos no tripulados y tecnologías de visión por computador para tareas de salvamento y rescate, buscando formas humanas, analizándolas y incluso dando información acerca de su estado vital. Sí esta tecnología se logra desarrollar e implementar en aeronaves con suficiente potencia para transportar a un ser humano, la capacidad de estos vehículos ya no solo se vería reducida a realizar una tarea de búsqueda y envió de información, si no que podría ampliarse a realizar la recogida y salvamento de la persona para su transporte a una zona segura o un hospital. Una flota de este tipo de aeronaves seria de maravillosa utilidad en casos de grandes catástrofes naturales.

Por último, a nivel personal, este trabajo me ha servido para profundizar y ampliar mis conocimientos en el campo de la visión por computador, y en concreto en la obtención y comparación de características entre dos imágenes, con el fin de realizar el reconocimiento de un patrón predeterminado. También me ha servido para profundizar en los conocimientos adquiridos en la asignatura de "Aplicaciones de la automática en vehículos". Ante todo se puede concluir que este trabajo me ha servido para aprender a redactar y a manejar un proyecto de esta importancia y envergadura, a la vez que a madurar personal y profesionalmente.

Capítulo 8: PRESUPUESTO

8.1 Costes de material

En la siguiente tabla se recogen los costes debidos al material utilizado para el desarrollo del trabajo

Tabla 5: Costes de material

Código	Unidad	Descripción	Medición	Precio	Total
1.1	Ud.	**Ardrone 2.0** Vehículo aéreo no tripulado de 4 rotores eléctricos en distribución de cuadricóptero. Desarrollado y distribuido por Parrot.	1	312,55 €	312,55 €
1.2	Ud.	**Ordenador** Computadora con procesador intel i7 o equivalente.	1	700,00 €	700,00 €
1.3	Ud.	**Visual Studio** Licencia Visual Studio 2013 para uso universitario.	1	0,00 €	0,00 €
			Total de la partida		**1012,55 €**

8.2 Costes de personal

En la siguiente tabla se recogen los costes de personal necesario para llevar a cabo este trabajo.

Tabla 6: Costes de personal

Código	Unidad	Descripción	Medición	Precio	Total
2.1	Meses	**Ingeniero industrial electrónico** Ingeniero industrial electrónico con conocimientos básicos de visión por computador, odometría visual y programación en C#.	2.5	1600,00 €	4000,00 €
			Total de la partida		**4000,00 €**

8.3 Presupuesto total

En la siguiente tabla se recoge el resumen de todas las partidas de este proyecto para formar el presupuesto total.

Tabla 7: Presupuesto total

Código	Unidad	Descripción	Medición	Precio	Total
1	Ud.	**Partida 1** Coste de material y equipos	1	1012,55 €	1012,55 €
2	Ud.	**Partida 2** Costes de personal	1	4000,00 €	4000,00 €
			Presupuesto total		**5012,55 €**

Capítulo 9: BIBLIOGRAFÍA

[1] A. de la Escalera, Visión por computador. Fundamentos y métodos, 2001.

[2] Military factory: http://www.militaryfactory.com/aircraft/unmanned-aerial-vehicle-uav.asp [Último acceso: Junio 2014].

[3] DefenseMediaNetwork: http://www.defensemedianetwork.com/stories/northrop-grumman-unmanned-aircraft-systems-achieve-100000-flight-hours-l-photos [Último acceso: Junio 2014].

[4] RQ-4 Global Hawk fact sheet: http://archive.today/7V1r [Último acceso: Junio 2014].

[5] Deagle.com MQ-1 Predator: http://www.deagel.com/Unmanned-Combat-Air-Vehicles/MQ-1-Predator_a000517002.aspx [Último acceso: Junio 2014].

[6] USAF MQ-1 factsheet: http://www.webcitation.org/66idKJIvU [Último acceso: Junio 2014].

[7] Cassidian Barracuda page: http://www.cassidian.com/en_US/web/guest/605 [Último acceso: Junio 2014].

[8] Air-Attack.com - EADS Barracuda: http://www.air-attack.com/page/75/Barracuda.html [Último acceso: Junio 2014].

[9] bga-aeroweb.com - MQ-8 Fire Scout (MQ-8B/C): http://www.bga-aeroweb.com/Defense/MQ-8-Fire-Scout.html [Último acceso: Junio 2014].

[10] Arming the Fire Scout – U.S. to Arm the MQ-8B with APKWS Guided Rockets: http://defense-update.com/20111109_arming-the-fire-scout-u-s-to-arm-the-mq-8b-with-apkws-guided-rockets.html#.U5mGPyiDpxM [Último acceso: Junio 2014].

[11] Honeywell: http://aerospace.honeywell.com/thawk [Último acceso: Junio 2014].

[12] Institute for Dynamic Systems and Control de Zurich: http://www.idsc.ethz.ch/ [Último acceso: Junio 2014].

[13] G. Bradski y A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 2008.

[14] S. E. Daniel Lélis Baggio, Mastering OpenCV with Practical Computer Vision Projects, 2012.

[15] Asociación Internacional de Vehículos No Tripulados: http://www.auvsi.org/home [Último acceso: Junio 2014].

[16] Especificaciones del Ardrone 2.0: http://ardrone-2.es/especificaciones-ar-drone-2/ [Último acceso: Junio 2014].

[17] web oficial de Microsoft C#: http://msdn.microsoft.com/es-es/library/kx37x362.aspx [Último acceso: Junio 2014].

[18] Programa DreamSpark de Microsoft: https://www.dreamspark.com/Student/Software-Catalog.aspx [Último acceso: Junio 2014].

[19] Microsoft Visual Studio: http://www.visualstudio.com/ [Último acceso: Junio 2014].

[20] Emgu CV: http://www.emgu.com/wiki/index.php/Main_Page [Último acceso: Junio 2014].

[21] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, pp. 1150–1157, 1999.

[22] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features,"in *In ECCV*, pp. 404–417, 2006.